FloatShield: An Open Source Air Levitation Device for Control Engineering Education ⋆

Gergely Takács, Peter Chmurčiak, Martin Gulan, Erik Mikuláš, Jakub Kulhánek, Gábor Penzinger, Marcel Vdoleček, Miloš Podbielančík, Martin Lučan, Peter Šálka and Dávid Šroba

Faculty of Mechanical Engineering, Slovak University of Technology in Bratislava, Námestie Slobody 17, 812 31 Bratislava, Slovakia (e-mail: gergely.takacs@stuba.sk).

Abstract: A novel reference design for an air levitation system to teach control engineering and mechatronics is introduced. The device is built as a swappable and compact extension shield for Arduino embedded microcontroller prototyping boards. The fully documented hardware design uses off-the-shelf electronic components and 3D printed mechanical parts to encourage easy and low-cost replication with editable design files provided online. The open source software includes an application programming interface for the Arduino IDE, MATLAB and Simulink. The examples included with the software demonstrate possibilities for typical classroom use.

Keywords: educational aids, control education, laboratory education, embedded systems, microprocessor control, Arduino, air levitation, air flotation, airflow, microcontroller

1. INTRODUCTION

Future control engineers and mechatronics professionals require years of intensive university training to be competitive in a volatile job market. Theoretical skills must be supplemented by hands-on experience in programming various hardware platforms and running closed-loop control experiments in the laboratory. Exposure to misbehaving "live" plants that operate and react unlike tidy simulations or textbook examples can be an eye-opening experience for many students.

Commercial manufacturers serve this niche by offering various laboratory trainers for control and mechatronics education. Their products are well made, come with supplementary software and even comprehensive course material for teachers. On the downside, the cost of a single device may rise to tens of thousands of Euros which may be prohibitive for many universities and laboratories across the world, especially if an entire classroom is to be equipped. Most devices require additional investment in the form of proprietary software and computing hardware. Cost is not the only drawback of commercial gear: students may not work unsupervised and taking equipment home for homework or long-term course projects is practically unimaginable.

Hence, numerous higher education institutions rely on custom-made laboratory equipment for teaching and some even for research. However, these one-off prototypes tend

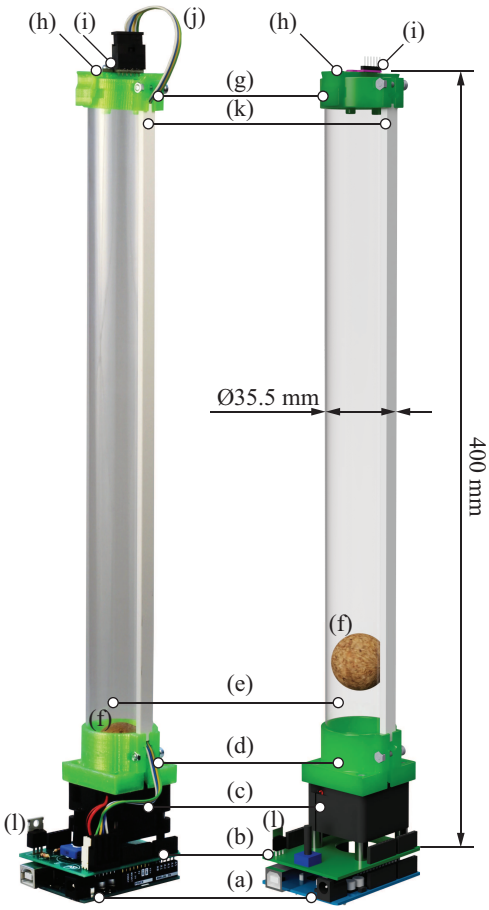


Fig. 1. Photograph of the FloatShield prototype (left) and its CAD render (right).

⋆ The authors gratefully acknowledge the contribution of the Slovak Research and Development Agency (APVV) under the contracts APVV-18-0023, APVV-17-0214 and APVV-14-0399. The authors appreciate the financial support provided by the Cultural and Educational Grant Agency (KEGA) of the Ministry of Education of Slovak Republic under the contract 005STU-4/2018.

Manuscript 1041 submitted to 21st IFAC World Congress, 2020.
Received November 7, 2019.

to stay within the confines of their birthplace; thus the hardware, software and course material only serves the needs of a limited group of people. Students cannot readily tap into external sources of information, researchers are unable to compare experimental results across unified hardware and outsiders cannot suggest improvements.

The success of the open-source Arduino microcontroller prototyping boards has seeped into university curricula across the world. The electrical connection layout of the boards is standardized and known as the Arduino R3 pinout, thereby the expansion of the base functionality of the microcontroller unit (MCU) is possible by the so-called "shields". A shield is essentially a printed circuit board (PCB) populated by a circuitry serving to add functionality such as Wi-Fi connection, motor control, GPS, etc.

In earlier work we have suggested to exploit the standardized nature, popularity and availability of this hardware to create laboratory experimental systems for closed-loop control that fit on a single Arduino shield. The thermal control of a 3D printer hotend was introduced in Takács et al. (2019a) while an optical feedback experiment costing less than 3 EUR to make was discussed in Takács et al. (2019b). We have laid out the framework of a novel class of educational aids for control and mechatronics in these previous contributions. These shall combine the low cost of custom-made devices, and the uniformity and refinement of commercial equipment by the help of open collaboration in hardware and software design. According to this, devices shall be (i) open-source, (ii) low-cost, (iii) simple, (iv) standardized, (v) well documented, (vi) compatible with the Arduino R3 electrical layout; furthermore shall (vii) fit on a single Arduino shield, (viii) use off-the-shelf components or 3D printing, (ix) need no ambiguous or improvised parts and (x) be easy to replicate and assemble by the average technically-apt person with minimal training.

This paper presents the well-known air levitation device (ALD) that is a familiar sight in some control laboratories in higher education. An air flotation device consists of a fan; a ball that is usually enclosed inside a tube and some means to measure its position. This system then enables one to control the altitude of the ball in closed loop control by varying fan power.

There are descriptions of several variations of ALD in the literature. The ball is often enclosed in a tube, while others forego this by relying only on the Coandă effect to restrict horizontal motion (Jernigan et al., 2009). Gauging the ball position can be troublesome: authors often choose infrared sensors for their low price and reliability (Barlas and Moallem, 2010; Chacon et al., 2017), using ultrasonic sensors are less precise and require large travelling distances (Dellah et al., 2000; Ovalle and Combita, 2019), choosing cameras increase cost and complexity (Escano et al., 2005; Visan et al., 2015), while industrial or laboratory grade laser sensors such in Schaefer et al. (2019) tend to be expensive and large. In most devices closed-loop control takes place on a personal or industrial computer, c.f. Escano et al. (2005); Jernigan et al. (2009), while Chołodowicz and Orłowski (2017) and Kuzhandairaj (2018) use an embedded device only to replace input-output modules. Microcontrollers are seldom used as primary computation

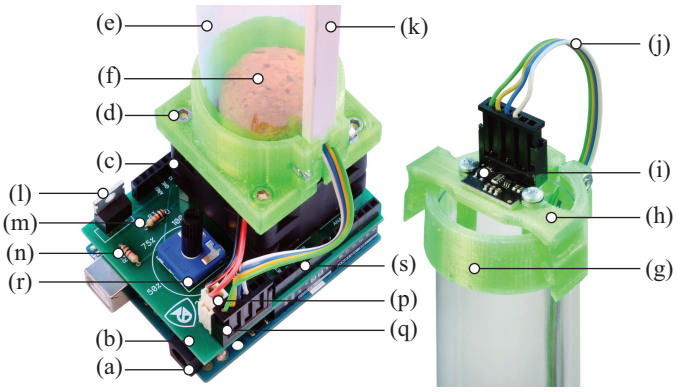


Fig. 2. Details at the bottom and top of the device.

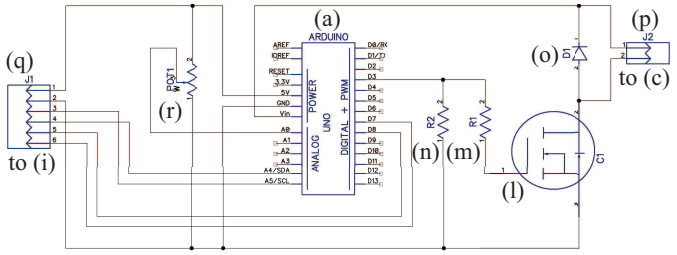
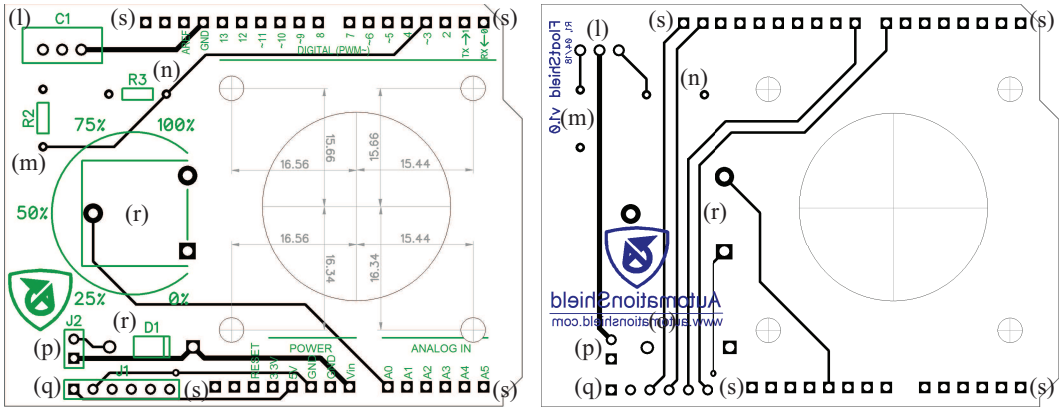


Fig. 3. Electrical schematics of the proposed device.

devices with the only exception being the now-outdated Basic Stamp considered by Dellah et al. (2000), while Chacon et al. (2017) and Saenz et al. (2017) favor more serious application-class embedded devices, such as the Raspberry Pi or the BeagleBone Black. The approximate price of the ALD is rarely discussed, however they range from ∼€ 70 for the system featured in Chołodowicz and Orłowski (2017) to ∼€ 135 in Dellah et al. (2000). As for size, several examples in the literature require considerable floor space in the laboratory or even hoists from the ceiling (Jernigan et al., 2009), while others are more than 2 m tall, as judged by the illustrations provided (Chołodowicz and Orłowski, 2017). Documentation is sparse at best, with the devices not designed to be replicated by others. A notable and interesting exception is in Chacon et al. (2017), where the authors have published the downloadable and editable 3D design files, PCB layouts and accompanying software. Nevertheless, this project seems to be abandoned for some years now. Another engaging project is discussed in Saenz et al. (2017), where the authors have embraced some of the open source hardware principles discussed earlier in this article (e.g. i–iii,x).

Here we propose a novel reference design for an open-source air levitator system built for Arduino prototyping boards that can be manufactured under € 30 with access to a 3D printer. The hardware—that we shall refer to as the "FloatShield" in the upcoming discussion—is part of a wider initiative to create affordable control engineering and mechatronics trainers. This paper introduces the hardware, software and examples for the FloatShield [1] .

Manuscript 1041 submitted to 21st IFAC World Congress, 2020.
Received November 7, 2019.



(a) Top layer. (b) Bottom layer.

Fig. 4. Printed circuit board with a 1:1 scale of the proposed device showing traces and pads (black), drilled holes (gray) and the silkscreen layers (green, blue).

2. HARDWARE

Figure 1 shows a photograph of the prototype on the left and a CAD file render on the right, while Fig. 2 features a detail of the top and the bottom of the device. The base of the FloatShield is an unmodified Arduino board (a) with the standard R3 pinout and 5 V logic levels, such as the Arduino Uno or Arduino Mega2560. The shield itself is then built on top of a PCB (b) to which the rest of the electronic components and mechanical parts are attached. A compact and powerful axial fan (c) is held in place by spacers, while its top is occupied by a 3D printed tube clamp (d). The clamp holds the bottom of a transparent tube (e) in which a ball (f) may move in the vertical dominant direction. The end of this tube is terminated by a 3D printed flange (g) that crossed by a sensor holder assembly (h). We created the 3D printed parts on a Prusa i3 MK3, all these are available for download both as CAD files and in printing-ready sliced formats [2]. Besides the fan, the second essential component of the FloatShield is the ST Microelectronics VL53L0X time of flight (TOF) pulsed infrared laser ranging sensor (i) providing accurate readings on the ball position inside the tube. A flat ribbon cable (j) runs through the side of the tube inside a U-profile (k) and is connected to the PCB at the bottom.

The electronic schematics of the FloatShield are shown in Fig. 3, while some parts are visible on the photograph in Fig. 2 as well. The TOF sensor and the fan are only represented by their connectors, while we assume that 12 V external power is drawn through the `VIN` pin of the Arduino. The fan is powered by an N-channel MOSFET (l), and driven by the `D3` PWM capable microcontroller pin through an 1 kΩ current limiting resistor (m). Floating states are handled by 10 kΩ pull-down resistor (n), while a diode (o) ensures back electromotive-force (EMF) protection. A connector (p) finally leads to the fan terminals. Since the fan requires 12 V and more current than the USB powered Arduino can handle, a separate wall adapter power supply is required to operate the device.

Since the TOF sensor is integrated to a convenient open-source breakout board, it can be effortlessly connected (q) to the I2C bus of the Arduino (`SDA`, `SCL`). Finally, the potentiometer (r) runner is attached to the `A0` ADC capable pin of the board. The shield is fixed mechanically and electrically to the Arduino board through stackable header pins (s). The location of the components is illustrated in Fig. 4 as well, featuring the 1:1 size PCB layout. The axial fan is mounted to the board on standoffs and a hole cut at the manufacturing stage of the PCB increases the air intake capacity.

The components necessary for this reference design are listed in Tab. 1. The total for the FloatShield excludes postage or labor and it comes under €30, making it affordable to laboratories with limited funding and even students. We have excluded the Arduino itself from this sum (€20 for original, €5 for third party) and the external power adaptor (∼€1–5), but even with these external parts the total price remains modest while these components can be re-used for other similar trainers, c.f. Takács et al. (2019a).

3. SOFTWARE

The proposed device is augmented by open-source software implementing a comprehensive API and examples for the Arduino IDE, MATLAB and Simulink. Since the FloatShield belongs to a wider effort to create open-source laboratory aids for control and mechatronics teaching, the collection of routines and examples for this particular system are a part of the "AutomationShield" library [3]. The goal of the API is to simplify working with various hardware trainers across different platforms, while retaining as much consistency as possible.

3.1 Arduino API

The AutomationShield library for the Arduino IDE behaves as any other similar module, and may be installed quickly even by inexperienced users using the graphical interface. The header `FloatShield.h` contains the forward declaration of the class `FloatClass` for hardware

[1] Watch a video introduction to the FloatShield at `https://www.youtube.com/watch?v=RHkqxbUVUrw&t`.

[2] `https://github.com/gergelytakacs/AutomationShield/wiki/FloatShield`

[3] `https://github.com/gergelytakacs/AutomationShield/`

Manuscript 1041 submitted to 21st IFAC World Congress, 2020.
Received November 7, 2019.

Table 1. Component list of the proposed device, without an Arduino board[a,b].

Symbol	Part	Description	Qty.	UP	Price (€)
(b)	PCB	FR4, 2 layer, 1.6 mm thick	1	0.45	0.45
(c)	Fan	Axial, 12 V, 40×40 mm, 24.0 CFM; e.g. Sunon PMD1204PQB1	1	12.61	12.61
(d)	Tube clamp	3D printed, 16 g filament, print time 2:24 h.	1	0.30	0.30
(e)	Tube	Clear, ϕ35.5 mm, wall approx. 0.6 mm, 0.4 m; e.g. no. 113816	0.4	10.92	4.37
(f)	Ball	Cork, ϕ30 mm; e.g. no. 108269	1	0.63	0.63
(g)	Tube flange	3D printed, 5.8 g filament, print time 57 min.	1	0.11	0.11
(h)	Sensor holder	3D printed, 4 g filament, print time 43 min.	1	0.08	0.08
(i)	Sensor	ST Microelectronics VL5310X TOF sensor on a breakout board	1	5.47	5.47
(j)	Wire	∼0.5 m, 4 lead, 0.15 mm^2, multi-conductor ribbon; e.g. VFL 4x0,14	0.5	0.28	0.14
(k)	Cable shaft	U-shape, 8×330 mm, ASA polymer; e.g. 11796	1	1.55	1.55
(l),C1	MOSFET	IRF520, TO-220AB, e.g. IRF520NPBF	1	0.41	0.41
(m), R1	Resistor	1 kΩ , 2.5×6.8 mm, THT	1	0.01	0.01
(n), R2	Resistor	10 kΩ , 2.5×6.8 mm, THT	1	0.01	0.01
(o), D1	Diode	1N4001, e.g 1N4001-DCO	1	0.03	0.03
(p)	Connector (fan)	2×1pin, 0.1" pitch; e.g. TE Connectivity 280358	1	0.04	0.04
(p), J2	Jumper (fan)	2×1pin, 0.1" pitch; e.g. TE Connectivity 280370-2	1	0.16	0.16
(q)	Connector (sensor)	6×1pin, 0.1" pitch; e.g. TE Connectivity 280360	1	0.07	0.07
(q), J1	Jumper (sensor)	6×1 pin, 0.1" pitch; e.g. TE Connectivity 280372-2	1	0.36	0.36
(q),(p)	Connector pins	e.g. TE Connectivity 182206-2	14	0.06	0.90
(r)	Turning knob	5×18.7mm; e.g. ACP 14187-NE	1	0.09	0.09
(r), POT1	Potentiometer	10 kΩ	1	0.28	0.28
(s)	Header	10×1 pin, female, long / stackable, 0.1" pitch	1	0.06	0.06
(s)	Header	8×1 pin, female, long / stackable, 0.1" pitch	2	0.09	0.18
(s)	Header	6×1 pin, female, long / stackable, 0.1" pitch	1	0.09	0.09
-	Bolts	DIN 912 M3×40	4	0.10	0.41
-	Bolts	DIN 912 M3×16	2	0.04	0.08
-	Nuts	DIN 934 M3	6	0.03	0.16
-	Screws	DIN 7981F 2.9×9.5	2	0.03	0.05
-	Washers	DIN125 A 3.2×7×0.5 Polyamide washers	4	0.01	0.03
-	Standoffs	TFM-M3/10	4	0.12	0.46
				Total:	**€ 29.58**[a,b]

[a] For low quantity orders.
[b] Excluding labor and postage.

interface methods. An object called `FloatShield` is initialized from within this header by default. After including the header, the system can be simply started by calling the

```
FloatShield.begin();
```

method launching the I2C interface and starting the TOF sensor itself. In addition to this, the ranging timing budget is set to achieve fast sampling in closed loop, along with declaring certain working parameters.

Although the sensor provides distance readings directly in millimeters, the outputs should be scaled to the range of 0–100% in order to use the Serial Plotter functionality, where all outputs are scaled to the same axis. The calibration procedure is called by

```
FloatShield.calibrate();
```

finding the minimal and maximal distance readings and map these values to percentages. Later on, one may check whether the calibration procedure was invoked or not by the `wasCalibrated()` method and access the minimal and maximal distance of the ball from the sensor by `returnMinDistance()` and `returnMaxDistance()`.

The position of the ball is read by

```
y=FloatShield.sensorRead();
```

returning the scaled distance inside the tube as a floating point value from within the range of 0–100 %. Alternatively, `sensorReadDistance()` reports the distance between the sensor and the ball in millimeters, while `sensorReadAltitude()` gives its altitude relative to ground.

By supplying the input power `u` in the range of 0–100% to the

```
FloatShield.actuatorWrite(u);
```

method, the user can set the power sent to the fan through the power circuitry. This method checks for constraints to avoid overflow, maps the input to 8-bit PWM integers, then sends it to the `D3` pin of the Arduino.

Finally, user reference from the potentiometer is acquired by calling

```
r=FloatShield.referenceRead();
```

returning the position of the potentiometer runner as a floating point scaled to 0–100 %.

In addition to the hardware specific functionality, the AutomationShield library incorporates a straightforward interrupt-driven sampling framework available for multiple types of microcontroller baords, a PID routine and other functions to enhance the learning process even further.

3.2 MATLAB API

The naming convention of methods is kept consistent for MATLAB as well. The library is initialized once by running an installation script then, assuming that the MATLAB Hardware Support Package for Arduino is present, an object is created from the `FloatShield` class by the `FloatShield = FloatShield` statement.

Fig. 5. Algorithmic blocks for the FloatShield in Simulink.

Using MATLAB with Arduino does not compile m-script to hardware, thus invoking

```
FloatShield.begin('COM3', 'UNO');
```

will simply load a server code to the microcontroller, unless it is not already present. This means that the closed-loop control by the API in MATLAB is not real-time in the strictest sense, since commands are transferred through the serial link between the board and the computer and may be affected by transfer speed or operating system behavior. However, being able to use the high-level MATLAB script allows one to run live experiments under this already familiar software platform and, most importantly, to create and test more advanced control frameworks with minimal effort.

The operation of the MATLAB API is otherwise completely identical to the aforementioned Arduino version. Therefore methods such as `calibrate()`, `sensorRead()`, `actuatorWrite()` and `referenceRead()` are all implemented for the FloatShield in MATLAB.

3.3 Simulink API

Running a script installs the Simulink API similarly to the previously discussed MATLAB module. An installation of the Simulink Support Package for Arduino Hardware is required. The collection of the algorithmic blocks will be then available for use through the Simulink Library (see Fig. 5) and may be combined with all the other available blocks to create feedback control applications. The FloatShield unit retains the naming convention and features for individual input and output blocks (`Actuator Write`, `Sensor Read` and `Reference Read`) and a comprehensive block representing the entire device is also available (`FloatShield`).

4. EXAMPLES

This section demonstrates typical fundamental classroom examples for undergraduate-level control and mechatronics education that are included with the API. Let us note that a more advanced algorithmic treatment of the FloatShield is outside the limited scope of this article, however, the reader shall note that neither the hardware, nor the software interface prohibits educators to use it to present concepts such as adaptive, predictive and robust control.

4.1 System Identification

FloatShield represents a single-input single-output (SISO) system, where the manipulated input is voltage of the fan, $u(t)$, which translates into the airspeed inside the tube, $v(t)$, and the controlled variable is the vertical position of the ball, $h(t)$, which is directly measured by the TOF sensor. The fan coupled to the tube generates

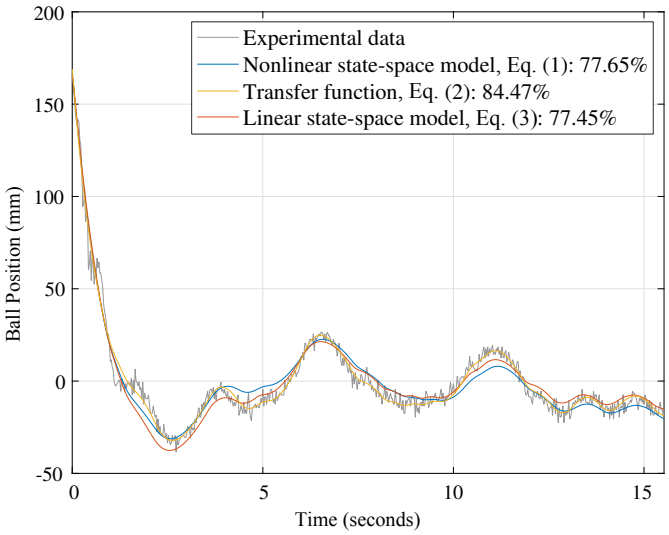


Fig. 6. Simulated response comparison of grey-box models – nonlinear state-space model (blue), linear transfer function (yellow) and linear state-space model (red) – with measurement data (grey).

a forced vertical air-flow that compensates the effect of gravity. These facts can be used to formulate a state-space representation, c.f. Chaos et al. (2019), as follows:

$$\dot{x}_1(t) = x_2(t), \tag{1a}$$

$$\dot{x}_2(t) = \frac{1}{2m} c_\mathrm{d} \rho A \left(x_3(t) - x_2(t)\right)|x_3(t) - x_2(t)| - g, \tag{1b}$$

$$\dot{x}_3(t) = \frac{K u(t) - x_3(t)}{\tau_1}, \tag{1c}$$

by assuming $x_1(t) = h(t)$, $x_2(t) = \dot{h}(t)$ and $x_3(t) = v(t)$, where m and A denote, respectively, the ball's mass and area exposed to the air flow, ρ is the density of air, g is the gravitational acceleration, c_d is the drag coefficient (assumed constant within the operation range), K the fan's gain and τ_1 time constant of the fan. While the values of the first four parameters are known a priori or can be determined in a reliable manner, the values of the other four paramters are unknown – hence need to be estimated. The nonlinear state-space model (1) can be expressed as a MATLAB ODE function, which can be then utilized in a grey-box identification procedure. The ODE model of the system in `FloatShield_ODE.m` is a part of the API.

Alternatively, one may opt for a linear(ized) system model in form of a transfer function

$$\frac{\delta H(s)}{\delta U(s)} = \frac{1}{s(\tau_1 s + 1)(\tau_2 s + 1)}, \tag{2}$$

where δ denotes perturbations of respective variables from the equilibrium point, c.f. Chacon et al. (2017), and $\tau_2 = (v - \dot{h})/2g$; or in form of a simple linear state-space model

$$\delta\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{1}{\tau_2} & \frac{1}{\tau_2} \\ 0 & 0 & -\frac{1}{\tau_1} \end{bmatrix} \delta x(t) + \begin{bmatrix} 0 \\ \frac{K}{\tau_1} \\ 0 \end{bmatrix} \delta u(t). \tag{3}$$

A worked example `FloatShield_Ident_Greybox.m` takes the experimental data and searches for the unknown parameters of the first-principle model (1). First, an initial guess of the system model is created based on assumed

Manuscript 1041 submitted to 21st IFAC World Congress, 2020.
Received November 7, 2019.

Table 2. Parameter values and initial guesses for the FloatShield's first-principle based grey-box model (fixed – known, free – estimated).

Parameter	Value
Ball's mass (m)	3.84E−3 kg (fixed)
Ball's exposed area (A)	5.7E−3 m^2 (fixed)
Drag coefficient (c_d)	0.74 (free)
Density of air (ρ)	1.23 kgm^{-3} (fixed)
Gravitational acceleration (g)	9.81 ms^{-2} (fixed)
Fan's gain (K)	7.78 $ms^{-1}V$ (free)
Fan's time constant (τ_1)	0.14 s (free)
Ball's time constant (τ_2)	0.6 s (free)

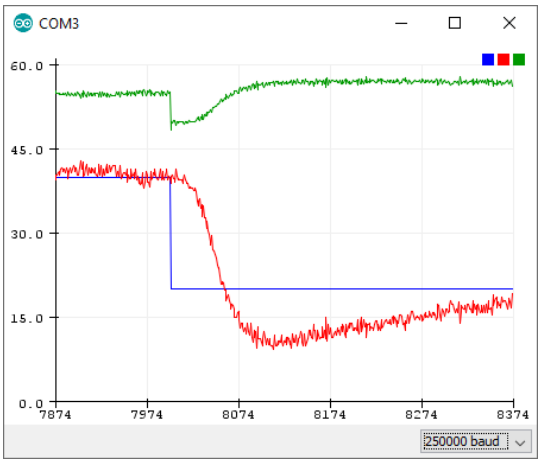


Fig. 7. Screenshot of closed-loop control in the Arduino IDE Serial Plotter (blue – reference, red – output, green – input). The vertical axis is fan power (%) and position (%).

parameter values and model structure. The unknown parameters are then found by a grey-box estimation procedure using an appropriate search method implemented in MATLAB's System Identification Toolbox. The resulting models provide a very good match to the measured input-output data (from 77 % to 85 %) as indicated in Fig. 6. The parameters used in the aforementioned models, their units and initial guesses are listed in Table 2.

4.2 Feedback Control

We shall demonstrate the closed-loop behavior of the device under PID control, since it most certainly forms an important part of any fundamental control or mechatronics course. The PID controller assumed in these demonstrations was hand-tuned ($K_P = 0.25$, $K_I = 5$ and $K_D = 0.01$), sampled at 25 ms and features hard input saturation and basic integral-windup handling by clamping.

Experiments may be designed and executed entirely in the Arduino IDE, since the Serial Plotter offers a convenient albeit simplistic quasi-real-time visual feedback of system response. The `FloatShield_PID.ino` demonstration example is included with the AutomationShield library and uses the sampling and PID framework from AutomationShield to aid the learning progress of beginners even further. Since it is not an option to scale the axes in the Serial Plotter, Fig. 7 only shows a short portion of the closed-loop response. The serial interface can be useful to export the data to an arbitrary terminal program or directly to

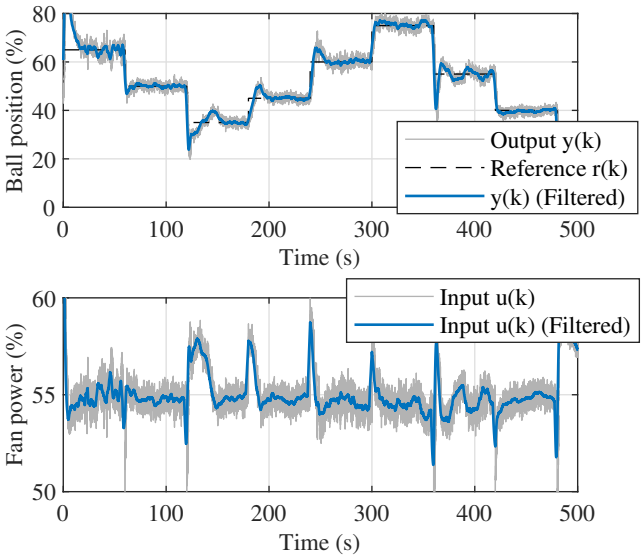


Fig. 8. Reference tracking by PID, created and executed in the Arduino IDE.

MATLAB. Figure 8 demonstrates position control of the ball gathered through serial logging software and displayed in MATLAB. Reference tracking is fairly accurate for the complex dynamics, while comparing inputs to the outputs exposes the nonlinearity of the system.

The application in `FloatShield_PID.m` implements the same example as introduced previously, but by using the MATLAB API. Sampling and computing the control decision is performed by the PC, while the Arduino only acts as an I/O interface. The response shown in Fig. 10 demonstrates that a consistent closed-loop behavior is expected even when resorting to MATLAB script. Moreover, as we indicated previously, this also means that students or researchers may make use of the immense power and high-level design possibilities of MATLAB.

Finally, Fig. 9 illustrates the Simulink scheme that can be literally created in seconds even by the most inexperienced beginners by using the FloatShield block as an input–output representation of the physical plant. Transcription to C/C++, compilation and uploading machine code to the board is handled by Simulink automatically, and the application runs stand-alone on the MCU while providing basic interaction with the host PC in the so-called External Mode.

5. CONCLUSION

We have presented an open-source reference design for an air floatation device that can be added to Arduino boards as an extension shield and can be manufactured under €30. The application programming interface for the Arduino IDE, MATLAB and Simulink retains compatible functional and naming conventions, thus students may conveniently switch between programming and development environments. Other devices featured in the AutomationShield library maintain this uniformity, therefore working with other types of physical plants takes less effort than normally. The low price, simplicity and

Manuscript 1041 submitted to 21st IFAC World Congress, 2020.
Received November 7, 2019.

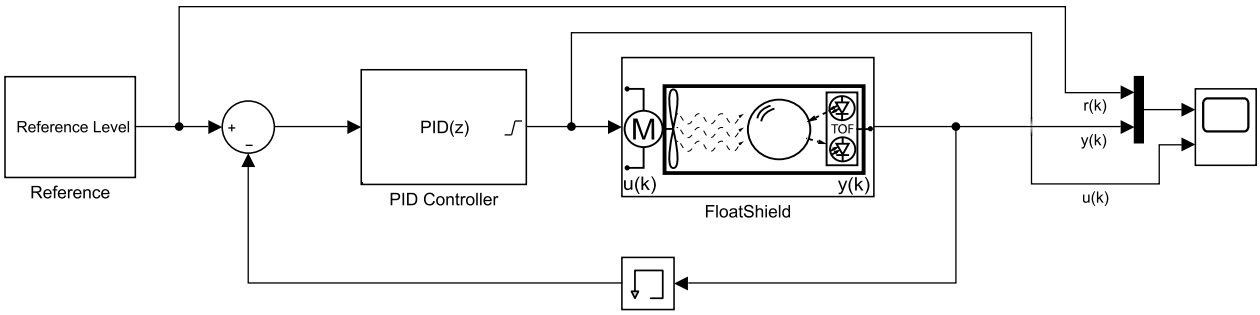


Fig. 9. Simulink scheme for the PID control of the FloatShield.

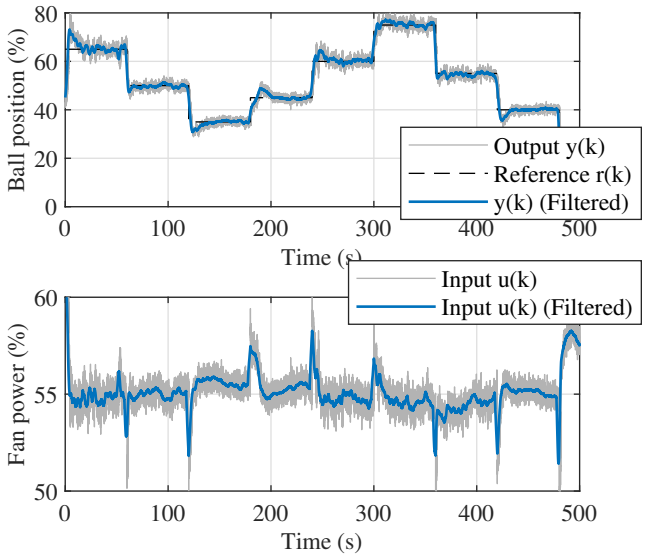


Fig. 10. Pseudo real-time PID control, created and executed in MATLAB script.

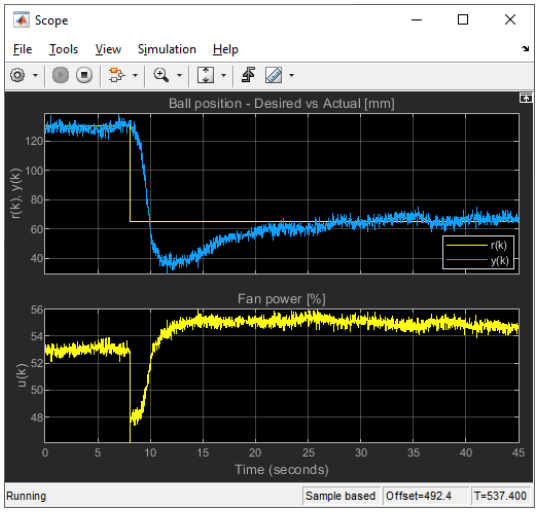


Fig. 11. Screenshot of live PID control on a Simulink scope.

openness of the FloatShield renders it a viable alternative to commercial trainers, enabling students to create and run control feedback experiments outside the laboratory.

There are several improvements that can be implemented in a future version of the FloatShield device. Just to name a few, the system should be made compatible with 3.3 V ARM Cortex M boards such as the Zero, size could be further reduced or, alternatively, a choking valve might be added to introduce a further input, c.f. Rover and Niemi (2002).

REFERENCES

T. Barlas and M. Moallem. *Developing FPGA-based Embedded Controllers Using Matlab/Simulink*, chapter 27, pages 543–556. IntechOpen, Rijeka, Croatia, 2010.

J. Chacon, J. Saenz, L. Torre, J. M. Diaz, and F. Esquembre. Design of a low-cost air levitation system for teaching control engineering. *Sensors*, 17(10), 2017.

D. Chaos, J. Chacón, E. Aranda-Escolástico, and S. Dormido. Robust switched control of an air levitation system with minimum sensing. *ISA Transactions*, 2019. In press, refer to `10.1016/j.isatra.2019.06.020`.

E. Chołodowicz and P. Orłowski. Low-cost air levitation laboratory stand using MATLAB/Simulink and Arduino. *Pomiary Automatyka Robotyka*, 4(21):33–39, 2017.

A. Dellah, P. Wild, and B. Surgenor. A laboratory on the microprocessor control of a floating ping pong ball. In *Proceedings of the ASEE Annual Conference & Exposition*, pages 5.32.1–5.32.8, St. Louis, Missouri, June 2000.

J. M. Escano, M. G. Ortega, and F. R. Rubio. Position control of a pneumatic levitation system. In *2005 IEEE Conference on Emerging Technologies and Factory Automation*, volume 1, pages 523–528, Sep. 2005.

S. R. Jernigan, Y. Fahmy, and G. D. Buckner. Implementing a remote laboratory experience into a joint engineering degree program: Aerodynamic levitation of a beach ball. *IEEE Transactions on Education*, 52(2): 205–213, May 2009.

J. C. Kuzhandairaj. Development,control and testing of an air levitation system for educational purpose. Master's thesis, Politecnico Milano, Milan, Italy, 2018. 854576.

D. M. Ovalle and L. F. Combita. Engaging control systems students with a pneumatic levitator project. In *2019 IEEE Global Engineering Education Conference*, pages 1093–1099, April 2019.

D. T. Rover and A. D. Niemi. Senior design as an agent for change in engineering education. In *32nd Annual Frontiers in Education*, volume 2, pages F3D–2–F3D–7, Nov 2002.

J. Saenz, J. Chacon, L. Torre, and S. Dormido. An open software - open hardware lab of the air levitation

system. *IFAC-PapersOnLine*, 50(1):9168–9173, 2017. ISSN 2405-8963. 20th IFAC World Congress.

M. Schaefer, D. Escobar, and H. Roth. Nonlinear identification and controller design for the air levitation system. In *2019 22nd International Conference on Control Systems and Computer Science*, pages 18–23, May 2019.

G. Takács, M. Gulan, J. Bavlna, R. Köplinger, M. Kováč, E. Mikuláš, S. Zarghoon, and R. Salíni. HeatShield: a low-cost didactic device for control education simulating 3D printer heater blocks. In *Proceedings of the 2019 IEEE Global Engineering Education Conference*, pages 374–383, Dubai, United Arab Emirates, April 2019a.

G. Takács, T. Konkoly, and M. Gulan. Optoshield: A low-cost tool for control and mechatronics education. In *Proceedings of the 12th Asian Control Conference*, pages 1001–1006, Kitakyushu-shi, Japan, Jun 2019b.

D. A. Visan, I. Lita, I. B. Cioc, and R. M. Teodorescu. A new approach for aerodynamic levitation based position control using digital image processing and data acquisition. In *2015 7th International Conference on Electronics, Computers and Artificial Intelligence*, pages AE–19–AE–22, June 2015.

Manuscript 1041 submitted to 21st IFAC World Congress, 2020.
Received November 7, 2019.